File Area

The ***<File_Area_Observational>*** class is a specific flavor of the more general *<File_Area>* parent class. It contains the essential file description class, *<File>*, followed by data structure descriptions. Each observational product *must* have at least one *<File_Area_Observational>*, and may have more than one if the core observational data are contained in multiple files.

There should be one *<File_Area_Observational>* for each data file. Each of these *File_Area_Observational* classes may contain multiple data structures, but you should *never* repeat the *<File_Area_Observational>* for each data structure in a single file. In other words, there should *never* be two *<File>* sub-classes with the same *<file_name>* value.

For additional explanation, see the PDS4 Standards Reference or contact your PDS node consultant.

Note that in the PDS4 master schema, all classes have capitalized names; attributes never do.

<File>

This class is handled identically in all the *File_Area_** classes. It is described in detail on the Filling Out the File Class page.

Data Structures

Following the *<File>* class are the data structure definitions. There is one of these for each data object in the file. You can list them in any order, though users generally prefer to see them in the same order in which they occur in the file.

As of this writing, these are the data structures allowed in an observational product. You should use the most specific data structure name that pertains to your data, to be able to take advantage of any transformation or analysis tools for that data type that would be applicable to your data.

Contact your PDS consultant if you don't see a suitable structure for your data to get advice on how to modify your data to fit archival structures. The data structures themselves are described on separate pages.

2D Array Structures

- **Array_2D_Image:** Use this for your standard 2D image data.
- **Array_2D_Spectrum:** Use this if your "image" is a 2D spectrum. In this case you will need to include the *<Spectral_Characteristics>* class from the *Spectral* discipline dictionary. See the Filling Out the Spectral Dictionary Class page.
- **Array_2D_Map:** Use this if your "image" is a map - especially if there is associated cartographic information.
- **Array_2D:** In general, you should use a more specific flavor of *Array_2D* when possible, to get access to additional support software where it exists and would be useful.

  Note: In just about all cases (and if you think you have an exception, contact your PDS consultant *immediately*), you will also need to include Display Dictionary classes to indicate how to draw or display these arrays so that geometric references and similar properties can be correctly tied to the display. See Filling Out the Display Dictionary Class for information on filling out these classes.

3D Array Structures

- **Array_3D_Image:** Use this for banded images that aren't spectral cubes (RGB color images, for example), or image stacks for which there is a cohesive third dimension. If your image stack is really a 2D image with a series of masks or backplanes, the masks/backplanes should generally be described individually (or as their own sequence).
- **Array_3D_Movie:** Use this structure for a time sequence of 2D images. Note that this is still an array structure, *not* a contemporary movie format like MPEG.
- **Array_3D_Spectrum:** Use this structure for spectral cubes to get access to transformation and analysis tools associated with this data type. In this case you will need to include the <*Spectral_Characteristics*> class from the *Spectral* discipline dictionary. See the Filling Out the Spectral Dictionary Class page.
- **Array_3D:** Use this generic 3D array structure if your data is sufficiently different from the other 3D array types that you don't need or want access to associated software for those types.

  Note: In just about all cases (and if you think you have an exception, contact your PDS consultant *immediately*), you will also need to include Display Dictionary classes to indicate how to draw or display these arrays so that geometric references and similar properties can be correctly tied to the display. See Filling Out the Display Dictionary Class for information on filling out these classes.

Fixed-Width Table Structures

- **Filling Out the Table_Character Data Structure:** Use this for fixed-width character tables. Note that for the time being, only ASCII characters are allowed in *Table_Character* data, so the fact that "width" is measured in bytes is equivalent to measuring it in characters.
- **Filling Out the Table_Binary Data Structure:** Use this for fixed-width binary data tables.

Variable-Width Table Structures

- **Filling Out the Table_Delimited Data Structure:** This structure is used when parsing rules must be followed in order to convert data in the input file into programmatic memory structures. In order to use this structure, the parsing rules must be documented, reviewed and accepted by PDS so that the data format can be supported by the standard PDS transformation tools. Contact your PDS node consultant for details.

  A *Table_Delimited* structure is used with the PDS/DSV parsing standard to define the inventory tables which list the products comprising a *Collection*.

Header Structures

- **Filling Out the Header Data Structure:** Use this for labelling FITS and VICAR headers in your data. If you have some other kind of header in your data, talk to your PDS node consultant about which data structure is appropriate.

Other Structures

- **Array:** The generic *Array* data object should only be used as a last resort. Contact your PDS node if you need to use this form of the data structure.

- **Array_1D:** The degenerate array is primarily for specialized fields and particles data. If you're working withe PPI node, they'll provide you with information and examples for this data structure. Otherwise, you should probably avoid it. When in doubt, check with your PDS consultant.
- **Stream_Text:** Parsable byte streams can be quickly and easily interpreted using standard rules. Common examples of parsable byte streams include:
    - text stored in records representing lines on a printed page
    - ASCII tabular data stored as records of variable length

  In these two examples, each record would end with a delimiter (for example, an ASCII carriage-return line-feed pair), making the byte stream parsable. PDS also recognizes SPICE kernels, certain header structures, and XML files as falling within the definition of a parsable byte stream.